3

The LSS-USDL Model

In Chap. 2, we studied four theories that provided a comprehensive view of service. This chapter starts by complementing the study made by looking into business model conceptualizations to create an evaluation framework that will help in identifying a set of concepts to be used for the creation of a service system model. Once the concepts are identified, they will be structured and organized into what we call a 6-point interaction star model. The model, called LSS-USDL, was implemented using semantic web technologies.

3.1 Service System Evaluation Framework

Related research has proposed several business model conceptualizations. We briefly present eight of these proposals that are relevant to our research as they define concepts that pertain to both external and internal views of service systems. We do not explain these conceptualizations in detail, but merely list concepts relevant to a service system model. It should be noted that these proposals are unrelated to the service theories reviewed in the previous section, hence, both types of related work will be used in the next section to derive the most common service system concepts.

3.1.1 Business Model Conceptualizations of Service Systems

Alt and Zimmermann [2] distinguished six generic elements as a comprehensive framework to develop *business models*: `Mission`, `Structure`, `Processes`, `Revenues`, `Legal issues` and `Technology`. Published in 2001, this is the earliest proposal in our study, but as we can see by analyzing Table 3.1 it already mentions most of the generic concepts that newer models used the most. This indicates that it had an impact in the field.

Petrovic et al. [32] divided a *business model* into seven sub-models: `Value model`, `Resource model`, `Production model`, `Customer relations model` (it was further divided into `Distribution model`, `Marketing model` and

`Service model`), `Revenue model`, `Capital model`, and `Market model`. The naming of this model's elements hints at a lower level description for each of them. However, the authors do not identify any further characteristics.

Kaner and Karni [20, 22] proposed CAIOPHYKE, a service model based on 9 major classes: `Customers`, `Goals`, `Inputs`, `Outputs`, `Processes`, `Human enablers`, `Physical enablers`, `Information enablers`, and `Environment`. Each of these major classes can be further described by main classes, which can then be further described by their respective minor classes. This model was developed based on a study with 150 student projects that covered around 100 service domains [21]. This is one of the most comprehensive models found in the literature. However, it features a high level of complexity without a proper formalization, which prevents from creating an abstraction to handle complexity.

In e^{3}service [23, 24], Kinderen and Gordijn focused on satisfying consumer needs and displaying the various value offerings from different services for an easier comparison. Therefore, the elements of this model are different from other approaches. This model is a valuable contribution to the state of the art as it is represented by a machine-readable ontology, the level of formality we envision for our model. However, its scope is customer-oriented, while we seek a manager-oriented approach that provides a view on how a service system operates.

Spohrer and Maglio [36] defined a service as value-cocreation and list ten related foundational concepts: `Ecology`, `Entities`, `Interactions`, `Outcomes`, `Value proposition based interactions`, `Governance mechanism based-interactions`, `Stakeholders`, `Measures`, `Resources`, `Access rights` and `Questions` [36]. Table 3.1 shows that it is one of the most complete models of our study.

Osterwalder and Pigneur [31] propose the Business Model Canvas, a high-level graphical tool for business modeling. The model uses the concepts `Value proposition`, `Customer segments`, `Channels`, `Customer relationships`, `Key activities`, `Key resources`, `Key partners`, `Cost structure`, and `Revenue Streams`. This model and its tool are very simple and easy to understand and enjoy some popularity.

Fielt [14] extended the Business Model Canvas by addressing its strongest limitations: the lack of partnering (c.f. [15]) and co-creation (c.f. [13]) concepts. This increased the complexity of the original model. However, Table 3.1 shows that this new model only contributes to one more element of the common concepts, so there is a risk that this increase in complexity might not be beneficial.

Zolnowski et al. [41] tried to tackle the issue of lack of elements of the original Business Model Canvas to describe co-creation. This proposed approach focuses on a redistribution of the elements and their connections, rather than changing them as seen in Fielt's approach. Hence, this model shares the same concepts as the original Business Model Canvas, but their organization changes (p.158 [41]).

3.1.2 Evaluation Framework

Comparing the related work reviewed in the previous chapter and section, it is possible to identify common concepts for describing service systems and, thus, derive a *service evaluation framework* of the most frequent and relevant concepts. The most common concepts identified are the `Goals`, `Stakeholders`, `Processes`, `Inputs`, `Outputs`, `Resources`, `Measures`, `Legal` and `Financial` (Table 3.1).

	Goals	Stakeholders	Processes	Inputs	Outputs	Resources	Measures	Legal	Financial
Vargo and Lusch (2004)	■	■	□	■	■	■		■	■
Sampson and Froehle (2006)	■	■	■	■	■	■			
Poels (2010)	■	■	■	■	■	■	□		■
Alter (2013)	■	■	■	■	■	■	■	■	
Alt and Zimmermann (2001)	■	■	■			□		■	□
Petrovic et al. (2001)	■		■			■			■
Kaner and Karni (2007)	■	□	■	■	■	■	□	□	□
Kinderen and Gordijn (2008)	■	■	□		■				□
Spohrer and Maglio (2009)	■	■	■	■	■	■	■	□	
Osterwalder and Pigneur (2010)	□	■	■			■			■
Fielt (2010)	□	■	■	■		■			■
Zolnowski et al. (2011)	□	■	■			■			■

Table 3.1. Service Model Evaluation Framework (empty = no contribution; □ = moderate contribution; ■ = important contribution).

`Goals` are one of the most used concepts in the studied models. There is no doubt that this is a critical element for a service model, not only because of its wide acceptance among the studied approaches, but also because it states the objectives of the service system and its value proposition to consumers.

`Stakeholders` are one of the most important concepts of a service, since it is conditioned by the people and organizations involved. This concept is used by almost all the studied approaches due to its importance. In most service models, there is an attribute for service customers. In the Business Model Canvas from Osterwalder [31] and the two studied improved approaches there is also an attribute for service partners [14, 31, 41]. Spohrer and Maglio [36] propose additional attributes which specialize stakeholders into authorities and competitors.

`Processes` are, along with `Goals`, a concept that all studied approaches share. This concept is of utmost importance when describing services from an internal organization, because corporations must have a strong knowledge of the processes needed for their services, to identify bottlenecks, and other issues.

`Inputs` are described in a small set of service models. Spohrer and Maglio [36] refer to them using the concept of `Ecology`. Fielt [14], when extending the Business Model Canvas, adds `Partner activities` and `Customer activities`, which act as an input for the service. Karni and Kaner's CAIO-PHYKE model [22] features the major class `Inputs`.

`Outputs` are also described in a small set of service models. Spohrer and Maglio [36] refer to them using the concept of `Outcomes`. e^{3}service [24] features outputs in the classes `Consequence`, `Benefit`, and `Value derivation`. Karni and Kaner [22] feature the major class `Outputs`.

`Resources` are described in most service description models, being absent just in e^{3}service. Alt and Zimmermann's approach [2] is the only model that does a partial description of this concept, focusing only on technology.

`Measures` refer to how the company can know its services' performance receive feedback of their operations. Only a small number of models were found in the literature that addressed this concept, as shown in Table 3.1.

`Legal` is the concept for the legal aspects of a service or business. It has a surprisingly low presence in the literature. Exceptions are Alt and Zimmermann [2] who propose `Legal issues` as one of their six generic elements of a business model; Karni and Kaner [22] use the main class `Legal factors` in the major class `Environment`; and Spohrer and Maglio identify `Governance mechanism based interactions` and `Access rights` [36].

`Financial` is the concept for the financial aspects of a service. This concept is used in most of the studied approaches. Hence, it is also an important concept for developing a comprehensive service model and evaluation framework.

3.2 Concepts and Building Blocks

The central concept of the service system model we propose is the notion of co-creation (which we will later call an interaction point). This concept shifts our study of economic activity from a Goods-Dominant logic (GD) where value exchange is perceived through goods transactions to a Service-Dominant logic (SD) where value exchange is co-created by all parties of service interactions [26]. Therefore, we no longer see value exchange as a provider delivering value to a customer by selling a product, but rather as both provider and customer co-creating value to each other during service interactions. Since co-creation during service interactions is a core feature of service systems and the interactions flow is also a core feature in service blueprints [34], we can conclude that a service system should be represented by its flow of interactions and their contextual information, such as the co-created value. Hence, we focus on describing service interactions, their context, and their flow.

The central concept of co-creation is complemented with a classification according to the interrogative pronouns commonly used in journalism: *what*, *how*, *where*, *who*, *when*, and *why*. It allows different people to look at the same service system from distinct perspectives by providing a holistic view on a system. The use of these pronouns has shown to be comprehensive for event-centered reporting [3]. This indicates that they may also be relevant to describe the events that are an integral elements of a service system. This strategy has shown to work well with the Zachman's framework for enterprise architecture [40] and other approaches by different authors in the field of information systems [8, 12, 35]. This classification enhances readability and understandability, gives an intuitive meaning to abstract concepts and helps organizations to ask questions about their processes and process models [35]. It also helps identifying some characteristics of a service offer and can be used as a common framework for querying different services [12].

Finally, the notion of co-creation and the interrogative pronouns are enriched with the concepts identified using the service model evaluation framework in the previous section. The framework combines the knowledge gathered by different authors in order to provide a set of concepts commonly used for the description of a service. The concepts are `Goals`, `Stakeholders`, `Processes`, `Inputs`, `Outputs`, `Resources`, `Measures`, `Legal`, and `Financial`.

One of our initial objectives was to avoid over-engineering the model. Thus, we followed a design philosophy which embraces the KISS principle[1] and parsimony to keep the final model simple. Our previous experience while developing the third version of USDL [4] showed us that a model which tries to capture all the details of a domain becomes expensive, large, and more complex than necessary which harms its adoption and understanding.

3.3 Model Structure

The central element of the model is an `Interaction`. By matching the framework of common concepts discussed in the previous section with the interrogative pronouns, we obtain the concept `Stakeholders` for the pronoun "who", the concept `Goals` for the pronoun "why", the concept `Resource` for the pronoun "what", and the concept `Process` for the pronoun "how". The interrogative pronouns "when" and "where" are easily matched with the spatial and temporal context, respectively, of a service interaction. Furthermore, for a service system analysis, we can study the stakeholders' participation based on the actual roles that take part of an interaction. In addition, the flow of different resources can also be matched with the concepts `Input` and `Output`. Hence, we can describe service interactions with the six interrogative pronouns by using the following concepts:

[1] KISS is an acronym and design principle for"Keep it simple, stupid" and was introduced by the U.S. Navy in 1960.

- Who: `Role` (stakeholder; human or computer actor)
- Why: `Goal` (a service interaction goal)
- What: `Resource` (may be physical, knowledge or financial)
- How: `Process` (the business process a service interaction belongs to)
- When: `Time` (expresses temporal dependencies)
- Where: `Location` (the locations where service interactions occur)

The resulting structure is called a 6-point interaction star model for describing service interactions, as shown in Fig. 3.1.

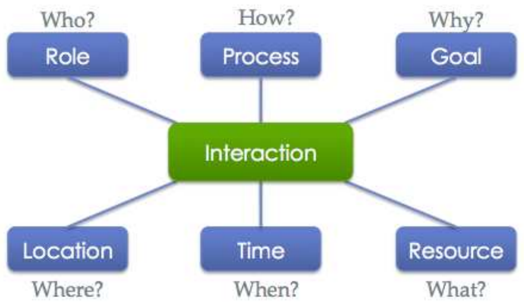

Fig. 3.1. 6-point interaction star model

Moreover, inspired by the work on service blueprinting [16], we may also classify interactions based on their area of action. A blueprint is a method created by Shostack [34] for analyzing a service delivery process by using a flow chart-like presentation to distinguish several types of customer interactions [25]. Thus, an interaction can be classified as a customer interaction, an onstage interaction, a backstage interaction, or a support interaction.

The foundational ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [28] classifies resources as endurants if they are physical objects or perdurants if they are not physical, such as services or events. Poels [33] classifies resources as operand if they are passive resources like objects or operant if they are knowledge and skills that embody competences. We can also find this pattern in some of the models we studied in the previous chapter. Therefore, resources should be classified as physical or knowledge. We also consider a third classification, financial resources, because of its importance for a business-oriented model.

Fig. 3.2 shows these extensions to the interaction and resource entities. Naturally, more extensions can be added to the model, for example, for domain specific modeling (e.g., e-government, IT services, consulting services, or e-banking).

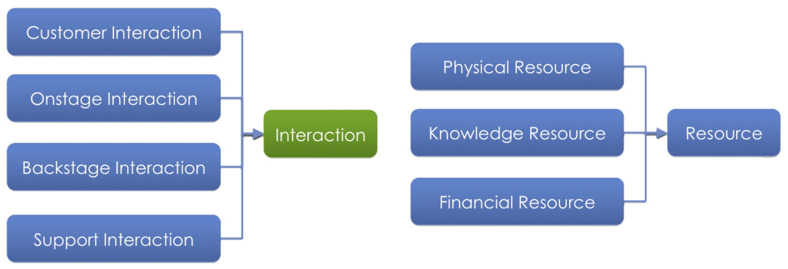

Fig. 3.2. Extensions to interaction and resource entities

3.4 Implementation Technologies

The implementation of the model was called Linked Service System for USDL (LSS-USDL) and it was guided by two main objectives: 1) to use semantic web technologies to make the model computer-understandable and sharable, and 2) to enable the model to refer to data from the Linked Data Cloud (LDC) [17].

By bridging LSS-USDL and the LDC, service systems can be semantically enriched by establishing meaningful relationships with data present in the LDC, which includes information such as company names, locations, and traded resources stored in semantic data sources such as DBpedia (`dbpedia.org`), GeoNames (`geonames.org`), and WordNet (`wordnet.princeton.edu`).

3.4.1 The Semantic Web

The World Wide Web Consortium (W3C) started to work on the concept of a Semantic Web with the objective of developing solutions for data integration and interoperability. The goal was to develop ways to allow computers to interpret (sometimes termed understand) information in the web. The Semantic Web identifies a set of technologies and standards that form the basic building blocks of an infrastructure that supports the vision of the meaningful web.

LSS-USDL is a service system description schema that was formalized using two technologies from the Semantic Web: the Resource Description Framework (RDF) [27] and RDF Schema (RDFS) [10]. RDFS was used to define a schema and vocabulary to describe services. This schema is used to create RDF graphs that describe individual services. Both, RDF and RDFS, are used by applications that need to interpret and reason about the meaning of information instead of just parsing data for display purposes. This section will provide an overview of the main frameworks, languages, technologies, and knowledge bases behind the Semantic Web, namely, RDF, RDFS, Turtle notation, SPARQL, and Linked Data. Nonetheless, it does not aim to provide a comprehensive description of these technologies. Thus, the reader is also refereed to the book *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL* [1].

3.4.2 RDF

The resource description framework was developed by the W3C to provide a common way to describe information so it could be read and interpreted by computer applications. It was initially designed using XML (eXtensible Markup Language [9]) as the underlying syntax, which enables syntactic interoperability. RDF provides a graph model for describing resources on the web. A resource is an element (document, web page, printer, user, etc.) in the web that is uniquely identifiable by a universal resource identifier (URI). A URI serves as a means for identifying abstract or physical resources. For example, `https://en.wikipedia.org/wiki/Incident_management` identifies the location from where a web page about the ITIL Incident Management service can be obtained and the following encoding `urn:isbn:1-420-09050-X` identifies a book using its ISBN.

The RDF model is based on the idea of making statements about resources in the form of a subject-predicate-object expression, a triple in RDF terminology. Each element has the following meaning:

Subject is the resource; the "thing" that is being described.
Predicate is an aspect about a resource and expresses the relationship between the subject and the object.
Object is the value that is assigned to the predicate.

RDF is based on a very simple data model based on directed graphs. A set of nodes are connected by (directed) edges. Nodes and edges are labeled with identifiers (i.e., URI) that makes them distinguishable from each other and allows for the reconstruction of the original graph from the set of triples. RDF offers a limited set of syntactic constructs – only triples are allowed.

Every RDF document is equivalent to an unordered set of triples, which describe a graph. For example, the RDF triple that describes the statement: "The goal of the ITIL Incident Management service is to solve incidents" is:

```
http://myitil.org/operation/IM_Service,
    http://w3id.org/lss-usdl/v1#hasGoal,
    http://myitil.org/operation/Solve_Incident
```

Listing 3.1. An RDF triple

The subject, `http://myitil.org/operation/IM_Service`, is a resource representing a particular ITIL service. This resource has the predicate (property) referenced by the URI `http://w3id.org/lss-usdl/v1#hasGoal` with the value `http://myitil.org/operation/Solve_Incident`. The statement can also be graphically represented as depicted in Fig. 3.3.

RDF blank nodes are used to express statements about individuals with certain properties without denominating the individual. The anonymity of blank nodes ensures that nothing besides the existence of the node can be

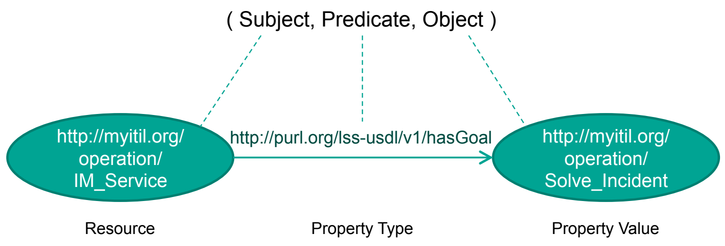
(Subject, Predicate, Object)
http://myitil.org/operation/IM_Service
http://purl.org/lss-usdl/v1/hasGoal
http://myitil.org/operation/Solve_Incident
Resource
Property Type
Property Value

Fig. 3.3. An example of an RDF graph

inferred. Blank nodes, as the name suggests, may only occur in the subject or object position of a triple.

Literals describe data values. They may only occur as property values. Literals are represented as strings. A shared interpretation is assumed to be given. Therefore, literals can be typed with a data type, e.g., using the existing types from the XML Schema specification. Untyped literals are interpreted as strings.

3.4.3 Turtle Syntax

While RDF is a data model, there are several serialization formats that can represent RDF graphs. Originally, XML was proposed and has been widely adopted by RDF data processing and management tools. It is noteworthy that the data model is not affected by the choice of any of the serialization formats; the graph structures remain unchanged. Turtle, the Terse RDF Triple Language, is one of the serializations. It is a compact syntax for RDF that allows representing graphs in natural text form [6]. It will be used in the remainder of this chapter.

In Turtle, every triple is completed by a full stop. A URI is represented in angle brackets and literals are enclosed in quotation marks. White spaces outside identifiers and literals are ignored. One way to represent the RDF statement from Fig. 3.3 using Turtle is shown in Listing 3.2.

```
<http://myitil.org/operation/IM_Service>
    <http://w3id.org/lss-usdl/v1#hasGoal>
    <http://myitil.org/operation/Solve_Incident> .
```

Listing 3.2. Turtle syntax representation of the RDF graph in Fig. 3.3

Turtle allows for abbreviation that further increase the readability. For example, multiple triples with the same subject or triples with same subject and predicate can be pooled as shown in Listing 3.3.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
3  @prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
4  @prefix myims: <http://myitil.org/operation#> .
5  @prefix lss-usdl: <http://w3id.org/lss-usdl/v1#> .
6
7  myims:IM_Service lss-usdl:hasGoal myims:Solve_Incident ;
8    rdf:type lss-usdl:ServiceSystem .
9
10 myims:Solve_Incident rdf:type lss-usdl:Goal .
11
12 myims:IMS12345 a myims:IM_Service ;
13   lss-usdl:Location [
14     geo:lat "48.7932" ;
15     geo:long "9.2258"
16   ] .
```

Listing 3.3. Turtle syntax representation of an RDF graph using abbreviations

The first lines introduce prefix abbreviations of the namespaces used. `rdf:type` (line 8) is a property to state that the resource `myims:IM_Service` is an instance of the class `myims:Service` system. The property `rdf:type` is often abbreviated to `a`. Capital first letters are used to indicate class names in contrast to individual and property names. The description of the location of the service `myims:IMS12345` makes use of a blank node representing the location resource. The location resource is not named but specified by its geographic coordinates embraced by square brackets.

3.4.4 RDF Schema

RDF Schema is a vocabulary language for RDF and allows to model vocabularies and ontologies. RDFS describes the logic dependencies among classes, properties, and values. While RDF provides universal means to encode facts about resources and their relationships, RDFS is used to express generic statements about sets of individuals (i.e., classes). RDFS associates resources with classes, states the relations between classes, declares properties, and specifies the domain and range of properties.

Classes in RDFS are much like classes in object oriented programming languages. They allow resources to be defined as instances of classes (by using the property `rdf:type`) and subclasses of classes. Subclass hierarchies can be specified by the RDFS property `rdfs:subClassOf`. The intuitive set theoretic semantics of class instances and subclasses (defined as member-of and subset-of relationships, respectively) ensures the reflexivity and transitivity of `rdfs:subClassOf`. The semantics of RDFS are specified in a W3C Recommendation [10].

Properties can be seen as attributes that are used to describe the resources by assigning values to them. RDF is used to assert property-related statements about objects, and RDFS can extend this capability by defining the class domain and the class range of such properties.

```
1  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
4  @prefix myims: <http://myitil.org/operation#> .
5  @prefix lss-usdl: <http://w3id.org/lss-usdl/v1#> .
6
7  myims:hasIncidentID rdf:type rdf:Property ;
8    rdfs:subPropertyOf ims:hasID ;
9    rdfs:label "Number required to uniquely identify an incident.
         This number should be used for all reference purpose both by
         internal and external stakeholders."@en ;
10   rdfs:domain myims:IncidentReport ;
11   rdfs:range myims:IncidentID .
12
13 myims:IM_Service lss-usdl:hasGoal myims:Solve_Incident ;
14   myims:implemented "1998-11-23"^^xsd:date .
```

Listing 3.4. Specification of domain and range of properties in RDFS

As the example shown in Listing 3.4 indicates, property hierarchies can be specified with the RDFS property `rdfs:subPropertyOf`. Literals, as shown in line 9 of Listing 3.4, describe data values for properties. A language tag, such as `@en` for English, is used to specify the language of the literal. Data type information can also be appended to literals (see line 14). Each data type is also identified by its URI, which in turn allows applications to interpret their meaning.

Given the logical statement nature of the knowledge represented with ontologies, traditional relational databases are not the ideal storage and query platform for RDFS. Knowledge is represented as sets of subject-predicate-object triples and these are most efficiently stored and accessed in dedicated triple stores, such as Jena TDB[2] and AllegroGraph[3]. Likewise, querying triple stores is done via specific query languages: the current standard language for querying RDF(S) is SPARQL [39].

3.4.5 Editors and Validators

Many tools have been developed to support users in modeling structured data, such as RDF and RDFS. Knowledge can be described with the support of ontology modeling tools like Protégé[4].

A traditional text editor can also be used to create service descriptions, but dedicated applications, such as TextMate for Mac, provide syntax highlighting

[2] Jena TDB `http://jena.apache.org/documentation/tdb/index.html`
[3] AllegroGraph `http://www.franz.com/agraph/allegrograph/`
[4] Protégé ontology editor and knowledge-base framework `http://protege.stanford.edu`

for Turtle, auto-completion, syntax validation, and format conversions. All helpful features that facilitate the modeling task.

RDF graphs can be validated against a schema and converted to different serialization formats (including RDF/XML, Turtle, and N3) with web-based tools like validators[5],[6] and translators [37].

3.4.6 SPARQL

The RDF information encoded is readable and interpretable by machines, e.g., software programs that utilize the knowledge in applications like a concert ticket selling application. SPARQL is a SQL-like query language that allows to retrieve data from RDF graphs. Answers are computed by matching patterns specified in a query against the given RDF graph.

Basic graph patterns are used in SPARQL queries when a set of triple patterns is matched. Listing 3.5 shows the SPARQL graph pattern query syntax. In SPARQL, Turtle is used to describe the graph patterns. In this example of a query, the set of artists, i.e., the individuals of the class `lss-usdl:ServiceSystem`, are retrieved and returned.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lss-usdl: <http://w3id.org/lss-usdl/v1#> .

SELECT ?service
WHERE
{
  ?service rdf:type lss-usdl:ServiceSystem .
}
```

Listing 3.5. A SPARQL query to retrieve instances of the class `ServiceSystem`

The answer of `SELECT` queries are bindings for the variables (denoted with a question mark) listed directly after the keyword `SELECT`. In the example, the query results in variable bindings for `?service`, which comprises, as shown in Table 3.2, a list of 3 service systems represented by their URI as used in the RDF graph. The `IM_Service` was already described. `EM_Service` is the Event Management service, a service to make sure services are constantly monitored, and to filter and categorize events in order to decide on appropriate actions. `PM_Service` is the Problem Management service, a service to manage the lifecycle of all problems and prevent incidents from happening.

Other query forms, e.g., `ASK`, `DESCRIBE`, and `CONSTRUCT`, allow to query for other kind of information. `ASK` returns a boolean answer about the existence of a solution for a specified graph pattern. A `DESCRIBE` query returns an RDF graph describing specified resources.

[5] http://www.rdfabout.com/demo/validator/
[6] http://www.w3.org/RDF/Validator/

ServiceSystem
`<http://myitil.org/operation/IM_Service>`
`<http://myitil.org/operation/EM_Service>`
`<http://myitil.org/operation/PM_Service>`

Table 3.2. Results of the SPARQL query shown in Listing 3.5

Linked Data

Linked Data [7] is a subset of the Semantic Web that adheres to the principles of the Semantic Web architecture: commitment to the use of RDF(S) and universal resource identifiers to denote "things". In particular, the following four design principles account for Linked Data:

- Use of URI to name things.
- Use of HTTP URI so that people can lookup the names.
- Lookups on those URI provide further information describing the things in RDF.
- Include links to other URI in the descriptions to allow people to discover further things.

The use of an HTTP URI allows machines and humans to lookup the name and get useful information about resources adhering to the RDF and SPARQL standards. The HyperText Transfer Protocol (HTTP) is prevalently used to exchange data in the web[7]. The use of an HTTP URI further guarantees the uniqueness of the identifier.

The resolvable resource description should contain links to other resource identifiers so that users can discover more things[8]. Linkage comprises external and internal links (for any predicate) and the reuse of external vocabularies, which can be interlinked. The special property `owl:sameAs` specifies the equivalence of different identifiers that refer to the same thing. For example, the Incident Management service is described in different vocabularies or websites. Overlapping data of different sources can be aligned by equivalence statements as illustrated in Listing 3.6.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<http://dbpedia.org/resource/Incident_management_(ITSM)> owl:sameAs
    <http://myitil.org/operation/IM_Service> .
```

Listing 3.6. Establishing the equivalence of resources using the property `owl:sameAs`

[7] See IEEE RFC2616 at http://tools.ietf.org/html/rfc2616 for details.
[8] Linked Data – Design Issues http://www.w3.org/DesignIssues/LinkedData

Adhering to the Linked Data principles has many advantages in the context of structured representation of data in the web but also in the context of the formal description of service systems. For example, for service search, selection, composition, and analysis.

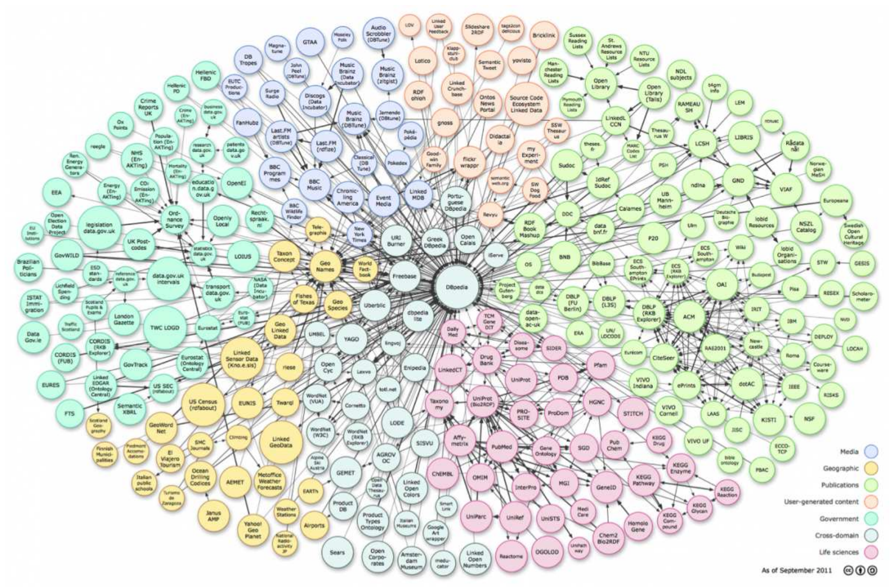

Fig. 3.4. The Linked Data cloud (http://lod-cloud.net/)

Fig. 3.4 shows a representation of the Linked Data cloud. The figure shows all the knowledge bases available on the web that can be remotely and programmatically accessed. The center of the giant interconnected network is DBpedia, a repository that contains the structured content from the information created as part of the Wikipedia project.

3.5 Model Implementation

Our idea behind the implementation of the 6-point interaction star model is pragmatic and it is based on the objective to create global service systems descriptions using computer-understandable descriptions.

3.5.1 Implementation Details

As explain in Sect. 3.4, the model was implemented as an RDF vocabulary, written in Turtle as opposed to XML due to its better readability [5]. To improve the integration with other semantic web initiatives, the model establishes links with various existing ontologies to reuse concepts from vertical and horizontal domains such as SKOS (taxonomies), Dublin Core (documents), FOAF (people) and so on.

The 6-point interaction star acts as the core of the model. A `ServiceSystem` class was used to group interactions.

A `Role` represents customers, managers, computer agents and so on. We link a role to its respective stakeholders with the property `belongsToBusinessEntity`. The property connects a `Role` to a `BusinessEntity` of the ontology GoodRelations. This ontology was chosen because it is widely accepted as a valuable Linked Data vocabulary for describing products and services [17].

The class `Process` represents an internal business process of the service system. It is particularly useful to filter interaction flows based on certain processes. Its usefulness can be improved by connecting it to modelled processes. Hence, we link it to a `Process` of the BPMN 2.0 ontology [30]. In future work, connections to different process modeling vocabularies may be considered, to expand the usefulness of this class.

The class `Goal` expresses a motivation for the occurrence of the interaction. This class is not connected to any element of the Linked Data Cloud because its meaning is contained in the context of its service system. Moreover, no relevant ontologies were found that could be used to extend the information of this class.

The class `Location` expresses where an interaction occurs. An instance of this element is connected to another through the property `isLocatedIn` to obtain a hierarchy level. This enables, for instance, associating an interaction with a room and finding that interaction when querying the room. It also has the property `isLocationFrom` that connects it to a `Feature` of the ontology Geonames [38]. This gives an unambiguous geographical context, since a Geonames `Feature` represents any city, country and so on and also uses a hierarchy level.

The concept `Time` gives a temporal context to interactions. It is connected to a `TemporalEntity` of the OWL-Time ontology [18]. This enables a high level of detail for temporal descriptions, such as the date and time of an interaction occurrence by using `DateTimeDescription` or its duration with the concept`DurationDescription`. It is also possible to define temporal relations between interactions by using properties such as `intervalBefore`, `intervalEquals` or `intervalAfter`. This enables a lightweight description of a process.

The class `Resource` captures inputs and outputs of the service system. Thus, an interaction can relate to a resource with the property `receivesResource` when it is being introduced from outside the service system; `createsResource` when it is created from within the service system; `consumesResource` when it is consumed from within the service system and `returnsResource` when it is provided to the outside of the service system. A resource is connected to `Quantitative Value` from the GoodRelations ontology so we may specify quantities. It can also be connected to `Resource` from DBpedia so that we may give it an unambiguous semantic element, i.e. a resource “Letter” might have an ambiguous meaning by itself (e.g., is it a mail letter or a letter from the alphabet?), but assigning it to a DBpedia `Resource` gives it an unambiguous semantic value.

As we previously discussed, `Interaction` and `Resource` also have subclasses. However, they are not mandatory, and other subclasses may be used instead. That is possible because they are subclasses of `Concept` from the SKOS ontology [19]. This means that they are concepts that can be extended by concept schemes [29]. Therefore, we can create a `ConceptScheme` from SKOS for `Interaction` and another for `Resource`, create their subclasses and add them to their respective concept schemes through SKOS property `hasTopConcept`. Similarly, if someone prefers a different set of subclasses, they may create a new concept scheme and assign the new subclasses as top concepts. This capability improves the model’s adaptivity and capacity to improve.

Listing 3.7 shows an extract of the RDF code of the LSS-USDL ontology.

```
# Every service system is defined by a lss-usdl:ServiceSystem class
lss-usdl:ServiceSystem a rdfs:Class, owl:Class;
    rdfs:label "ServiceSystem" .

# Every service system features a set of interactions
lss-usdl:Interaction a rdfs:Class, owl:Class;
    rdfs:subClassOf skos:Concept;
    rdfs:label "Interaction" .

# Every interaction relates to other entities, such as its location
lss-usdl:Location a rdfs:Class, owl:Class;
    rdfs:label "Location" .

# This property connects a service system to its interactions
lss-usdl:hasInteraction a rdf:Property;
    rdfs:label "has interaction";
    rdfs:domain lss-usdl:ServiceSystem;
    rdfs:range lss-usdl:Interaction .

# This property connects an interaction to its location
lss-usdl:hasLocation a rdf:Property;
    rdfs:label "has location";
    rdfs:domain lss-usdl:Interaction;
    rdfs:range lss-usdl:Location .

# A location can also be connected to an element of the Geonames
     ontology
lss-usdl:isLocationFrom a rdf:Property;
    rdfs:label "is location from";
    rdfs:domain lss-usdl:Location;
    rdfs:range gn:Feature .
```

Listing 3.7. LSS-USDL ontology RDF extract

3.5.2 Integration with the Linked Data Cloud

Another objective, no less important, was to integrate the model with the Linked Data Cloud. This means that the connection between entities of the LSS-USDL model must have a semantic meaning with entities of the LDC. The integration with the LDC is done by reusing relevant Linked Data ontologies, such as Geonames or DBpedia.

The LDC is generating tremendous interest and uptake by researchers and by the industry. The term refers to publicly available data on the World Wide Web in the form of knowledge represented by ontology languages like RDF and OWL, which are established standards by the W3C for metadata sharing and information integration [11].

Historically, corporate information describing data and services was closed inside private databases and "firewall". Linked Data is a recent movement which use Semantic Web advances to enable organizations to give a remote access of their internal data and service assets to others. For example, the US and UK governments already make their legislation available to citizens in a transparent manner using semantic languages. The set of all the datasets made accessible across the world is called the Linked Data Cloud. Driven by researchers, government agencies (e.g., `govtrack.us` and `legislation.gov.uk`), and companies (e.g., The Guardian and The National Library of Germany), the resulting Linked Data alone has grown to over 30 billion RDF triples.

However, in isolation the value of Linked Data is under-explored. By matching vocabularies defined by LSS-USDL and data of the LDC, we will be able to add background knowledge to service systems. For example, this integration enables to execute queries to find information about specific service resources annotated with DBpedia concepts (e.g., passport, medical record, and bill of materials). DBpedia is a repository of structured information retrieved for Wikipedia and accessible as RDF statements. As another example, it also enables to retrieve information, such as the country, population, postal code, and alternative names of the locations where services operate using GeoNames, an ontology with more than 8 million toponyms.

References

[1] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[2] Rainer Alt and Hans-Dieter Zimmermann. Preface: introduction to special section–business models. *Electronic Markets*, 11(1):3–9, 2001.

[3] Kevin Barnhurst and Diana Mutz. American journalism and the decline in event-centered reporting. *Journal of Communication*, 47(4):27–53, 1997.

[4] Alistair Barros, Uwe Kylau, and Daniel Oberle. Unified Service Description Language 3.0 (USDL) Overview, 2011.

[5] David Beckett and Tim Berners-Lee. Turtle-terse RDF triple language. *W3C Team Submission*, 14, 2008.

[6] David Beckett and Tim Berners-Lee. Turtle - terse RDF triple language. W3C team submission, W3C, March 2011. accessed Aug. 15, 2013.

[7] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[8] Andrew Blair, John Debenham, and Jenny Edwards. Requirements analysis for intelligent decision support systems. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 482–486. IEEE, 1994.

[9] Tim Bray, Jean Paoli, Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML) 1.1 (second edition). W3C recommendation, W3C, September 2006. accessed Aug. 15, 2013.

[10] Dan Brickley and Ramanathan Guha. RDF vocabulary description language 1.0: RDF Schema. W3C recommendation, W3C, February 2004. accessed Aug. 15, 2013.

[11] Jorge Cardoso. *The Syntactic and the Semantic Web*, pages 1–23. IGI Global, 2007.

[12] Marlon Dumas, Justin O'Sullivan, Mitra Heravizadeh, David Edmond, and Arthur ter Hofstede. Towards a semantic framework for service description. In Robert Meersman, Karl Aberer, and Tharam Dillon, editors, *DS-9*, volume 239 of *IFIP Conference Proceedings*, pages 277–291. Kluwer, 2001.

[13] Erwin Fielt. Alternative business model canvasses: A Partnering Canvas example. `http://fieltnotes.blogspot.pt/2010/12/alternative-business-model-canvasses.html`, 2010.

[14] Erwin Fielt. An Extended Business Model Canvas for Co-Creation and Partnering, 2010.

[15] Erwin Fielt. To what extent is the Business Model Canvas constraining? A Co-Creation Canvas example. `http://fieltnotes.blogspot.pt/2010/11/to-what-extent-is-business-model-canvas.html`, 2010.

[16] Sabine Fließ and Michael Kleinaltenkamp. Blueprinting the service company: Managing service processes efficiently. *Journal of Business Research*, 57(4):392–404, 2004.

[17] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[18] Jerry Hobbs and Feng Pan. Time ontology in OWL. *W3C working draft*, 27, 2006.

[19] Antoine Isaac and Ed Summers. SKOS Simple Knowledge Organization System Primer. W3C Working Group Note. *World Wide Web Consortium*, 2009.

[20] Maya Kaner and Reuven Karni. Design of service systems using a knowledge-based approach. *Knowledge and Process Management*, 14(4):260–274, 2007.

[21] Reuven Karni and Maya Kaner. Teaching innovative conceptual design of systems in the service sector. *Technological Forecasting and Social Change*, 64(2):225–240, 2000.

[22] Reuven Karni and Maya Kaner. An engineering tool for the conceptual design of service systems. *Advances in Services Innovations*, pages 65–83, 2007.

[23] Sybren Kinderen and Jaap Gordijn. e3service: An ontological approach for deriving multi-supplier IT-service bundles from consumer needs. In *Proceedings of the 41st annual Hawaii international conference on system sciences*, 2008.

[24] Sybren Kinderen and Jaap Gordijn. Reasoning about substitute choices and preference ordering in e-services. In *Advanced Information Systems Engineering*, pages 390–404. Springer, 2008.

[25] Holger Luczak, Christian Gill, and Bernhard Sander. Architecture for Service Engineering The Design and Development of Industrial Service Work. In Dieter Spath and Klaus-Peter Fähnrich, editors, *Advances in Services Innovations*, pages 47–63. Springer Berlin Heidelberg, 2007.

[26] Paul Maglio, Stephen Vargo, Nathan Caswell, and Jim Spohrer. The service system is the basic abstraction of service science. *Information Systems and e-business Management*, 7(4):395–406, 2009.

[27] Frank Manola and Eric Miller. RDF primer. W3C recommendation, W3C, February 2004. accessed Aug. 15, 2013.

[28] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb Deliverable D18, Ontology Library (final). *ICT Project*, 33052, 2003.

[29] Alistair Miles, Brian Matthews, Michael Wilson, and Dan Brickley. SKOS core: simple knowledge organisation for the web. In *International Conference on Dublin Core and Metadata Applications*, 2005.

[30] Christine Natschläger. Towards a BPMN 2.0 Ontology. In *Business Process Model and Notation*, pages 1–15. Springer, 2011.

[31] Alexander Osterwalder and Yves Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. Wiley, 2010.

[32] Otto Petrovic, Christian Kittl, and Ryan Teksten. Developing business models for ebusiness. *Available at SSRN 1658505*, 2001.

[33] Geert Poels. The resource-service-system model for service science. In *Advances in Conceptual Modeling–Applications and Challenges*, pages 117–126. Springer, 2010.

[34] Lynn Shostack. Designing services that deliver. *Harvard Business Review*, 62(1):133 – 139, 1984.

[35] Eva Söderström, Birger Andersson, Paul Johannesson, Erik Perjons, and Benkt Wangler. Towards a framework for comparing process modelling languages. In *Advanced Information Systems Engineering*, pages 600–611. Springer, 2006.

[36] Jim Spohrer and Paul Maglio. Service science: Toward a smarter planet. *Introduction to service engineering*, pages 3–30, 2009.

[37] Alex Stolz, Bene Rodriguez-Castro, and Martin Hepp. RDF translator: A restful multi-format data converter for the semantic web. Technical Report TR-2013-1, Universität der Bundeswehr München, July 2013.

[38] Bernard Vatant and Marc Wick. Geonames ontology. http://www.geonames.org/ontology, 2012. Accessed at 31/05/2013.

[39] W3C SPARQL Working Group. SPARQL 1.1 overview. W3C recommendation, W3C, March 2013. accessed Aug. 15, 2013.

[40] John Zachman. Enterprise architecture: The issue of the century. *Database Programming and Design*, 10(3):44–53, 1997.

[41] Andreas Zolnowski, Martin Semmann, and Tilo Böhmann. Introducing a Co-Creation Perspective to Service Business Models. In *Enterprise Modelling and Information Systems Architectures (EMISA)*, page 243, 2011.